数据结构与算法

Data Structrues and Algorithms

张岩

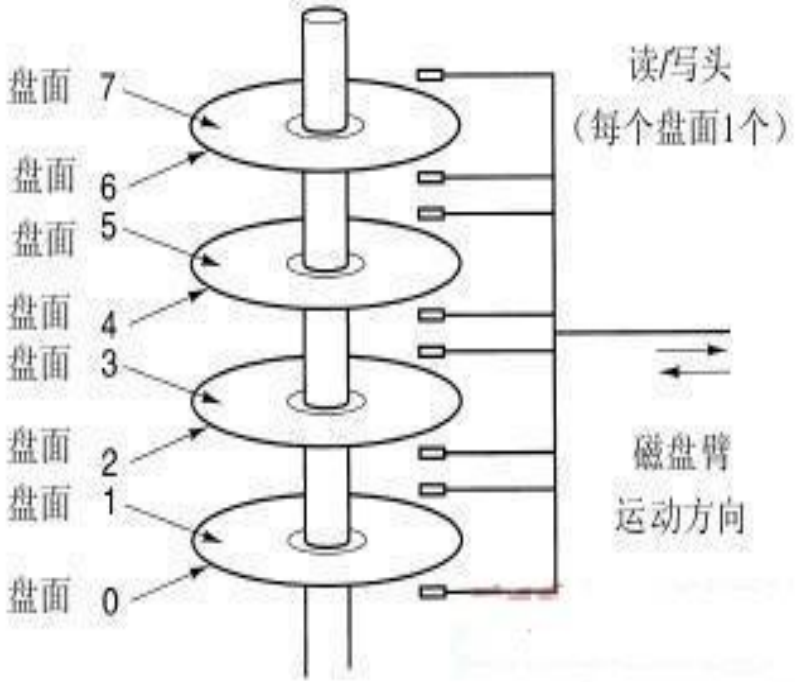
海量数据计算研究中心

哈工大计算机科学与技术学院

第7章 文件与外部排序

学习目标

- 掌握文件的相关概念；文件的各种组织方法及特点；查询、更新操作及其算法。
- 掌握外部排序的一般过程，熟练掌握适合外存特点的归并排序的相关技术。



磁盘驱动器的构造

本章主要内容

- 7.1 文件及文件操作
- 7.2 顺序文件
- 7.3 索引文件
- 7.4 ISAM和VSAM文件
- 7.5 直接存取文件（散列文件）
- 7.6 多关键字文件
- 7.7 磁盘文件的归并排序
- 7.8 磁带文件的归并排序
- 本章小结

7.1 文件及文件操作

相关概念

- 文件（FILE）是性质相同的记录组成的集合。习惯上称存储在主存储器（内存储器）中的记录集合为表，称存储在二级存储器（外存储器）中的记录集合为文件。
- 数据项：最基本的不可分的数据单位，是文件中可使用的数据的最小单位。
- 属性：记录中所有非关键字的数据项，称为记录的属性。
- 关键字、主关键字、次关键字

记录	学号	姓名	性别	年龄	数学	语文	物理	其它
A	003	张 三	男	18	90	80	80	
B	008	李 四	女	17	90	90	80	
C	009	王 五	女	19	89	70	93	
D	010	陈 中	男	19	66	77	68	
E	011	孙 二	男	18	91	88	78	
F	012	林 森	女	20	60	59	67	

7.1 文件及文件操作

文件的分类

- 按记录的类型不同可分成:
  - ①操作系统的文件：操作系统中的文件仅是一维的连续的字符序列，无结构、无解释。
  - ②数据库文件：数据库中的文件是带有结构的记录的集合。这类记录是由一个或多个数据项组成的集合，它也是文件中可存取的数据的基本单位。
- 按记录中关键字的多少，数据库文件可分成:
  - ①单关键字文件：文件中的记录只有一个唯一标识记录的主关键字。
  - ②多关键字文件：文件中的记录除了含有一个主关键字外，还含有若干个次关键字。
- 按记录含有信息的长度不同可分成:
  - ①定长记录文件：文件中每个记录含有的信息长度相同。
  - ②不定长记录文件：文件中每个记录含有的信息长度不等

7.1 文件及文件操作

文件的逻辑结构和物理结构

- 逻辑结构：是呈现在用户或应用程序员的记录间的逻辑关系，是用户对数据的表示和存取方式。着眼于用户使用方便。
- 物理结构：是数据在物理存储器上存储的方式，是数据的物理表示和组织。着眼于提高存储空间的利用率和减少存取记录的时间。
- 物理记录和逻辑记录之间可能存在下列3种关系：
  - ①一个物理记录存放一个逻辑记录;
  - ②一个物理记录包含多个逻辑记录;
  - ③多个物理记录表示一个逻辑记录。

7.1 文件及文件操作

文件的操作（运算）

- 文件的检索：
  - ①顺序存取：存取下一个逻辑记录。
  - ②直接存取：存取第i个逻辑记录。
    - ①和②两种存取方式都是根据记录序号(即记录存入文件时的顺序编号)或记录的相对位置进行存取的。
  - ③按关键字存取：给定一个值，查询一个或一批关键字与给定值相关的记录。数据库文件可以有如下4种查询方式：
    - 1. 简单询问：查询关键字等于给定值的记录。
    - 2. 区域询问：查询关键字属某个区域内的记录。
    - 3. 函数询问：给定关键字的某个函数。
    - 4. 布尔询问：以上3种询问用布尔运算组合起来的询问

7.1 文件及文件操作

文件的操作（运算）

- 文件的更新:
  - 文件的更新包括插入一个记录、删除一个记录和修改一个记录3种操作。
- 文件的操作方式
  - 可以有实时和批量两种不同方式。通常实时处理对应答时间要求严格，应在接受询问之后几秒钟内完成检索和修改，而批量的处理则不然。
  - 例如，银行的帐户系统需实时检索，但可进行批量修改，即可以将一天的存款和提款记录在一个事务文件上，在一天的营业之后再进行批量处理。
- 文件的组织方式
  - 顺序方式、索引方式、散列方式、链接方式

7.2 顺序文件

概念:

- 顺序文件（Sequential File）：是记录按其在文件中的逻辑顺序依次进入存储介质而建立的，即顺序文件中物理记录的顺序和逻辑记录的顺序是一致的。
- 连续文件：次序相继的两个物理记录在存储介质上的存储位置是相邻的顺序文件。
- 串联文件：物理记录之间的次序由指针相连表示的顺序文件

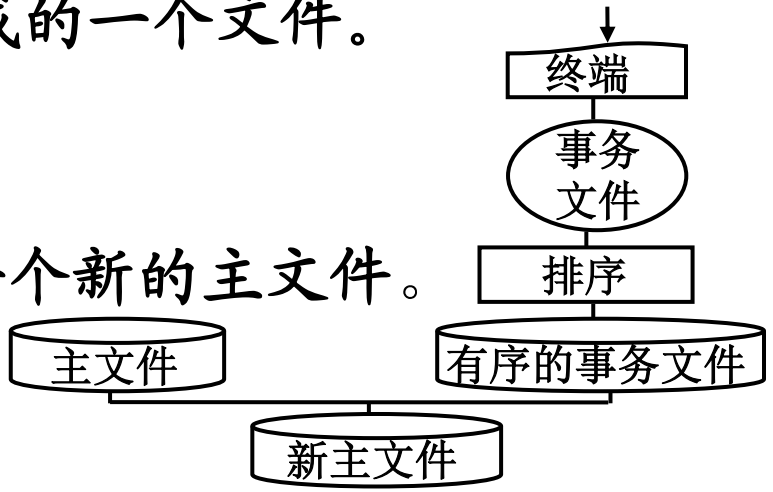
特点:

- 顺序文件是根据记录的序号或记录的相对位置来进行存取的文件组织方式。它的特点是:
  - ①存取第i个记录，必须先搜索在它之前的i-1个记录。
  - ②插入新的记录时只能加在文件的末尾。
  - ③若要更新文件中的某个记录，则必须将整个文件进行复制。

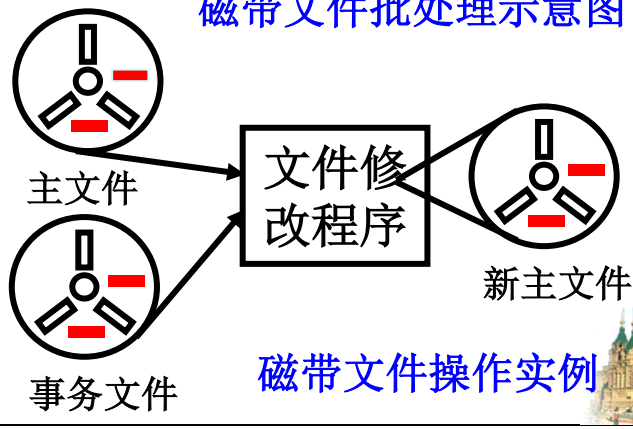
7.2 顺序文件

磁带上顺序文件的批处理

- 主文件：待修改的原始文件。
- 事务文件：所有的修改请求集中构成的一个文件。
- 磁带文件的批处理过程：
  - 首先对事务文件进行排序；
  - 然后将主文件和事务文件归并成一个新的主文件。
- 其中，
  - 主文件、事务文件和新的主文件分别存放中一台磁带上。
  - 主文件按关键字自小至大（或自大至小）顺序有序，
  - 事务文件必须和主文件有相同的有序关系。



磁带文件批处理示意图

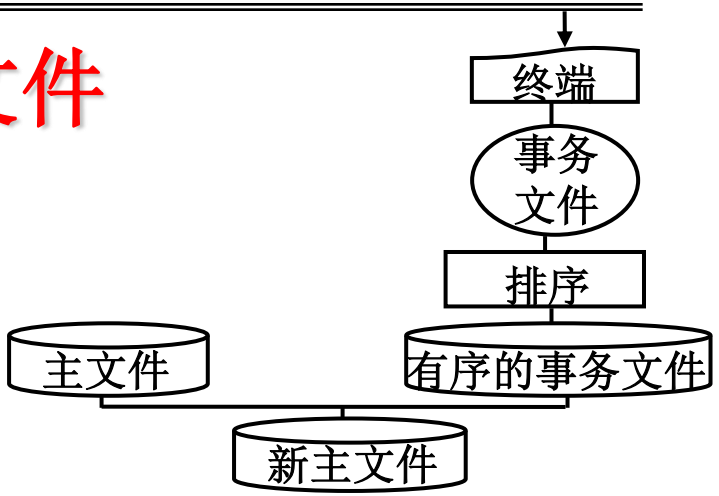


磁带文件操作实例

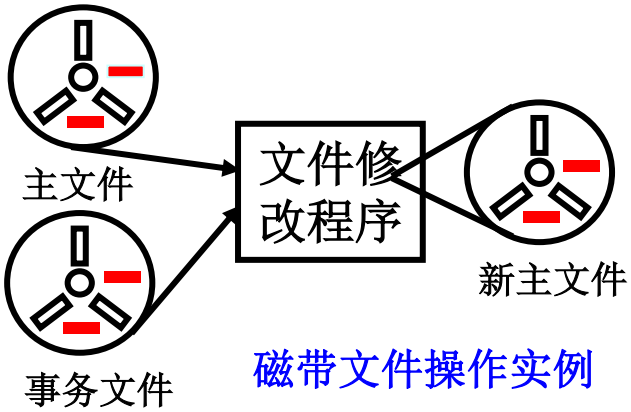
7.2 顺序文件

磁带上顺序文件的批处理—归并

在归并的过程中，顺序读出主文件与事务文件中的记录，比较它们的关键字并分别进行处理。对于关键字不匹配的主文件中的记录，则直接将其写入新主文件中。“更改”和“删除”记录时，要求其关键字相匹配。“删去”不用写入，而“更改”则要将更改后的新记录写入新主文件。“插入”时不要求关键字相匹配，可直接将事务文件上要插入的记录写到新主文件的适当位置。



磁带文件批处理示意图



磁带文件操作实例

磁盘上顺序文件的批处理

- 磁盘上顺序文件的批处理和磁带文件类似，只是当修改项中没有插入，且更新时不增加记录的长度时，可以不建立新的主文件，而直接修改原来的主文件即可。显然，磁盘文件的批处理可以在一台磁盘组上进行。

7.3 索引文件

- 基本术语:
  - 索引表：除了文件本身（称做数据区）之外，另建立的一张指示逻辑记录和物理记录之间一一对应关系的表。
  - 索引项：索引表中的每一项，由记录的关键字和记录的存放地址构成，总是按关键字（或逻辑记录号）顺序排列。
  - 索引文件：把索引表和主文件总称为索引文件(Indexed File)
  - 索引顺序文件：数据区中的记录也按关键字顺序排列的文件
  - 索引非顺序文件：数据区中的记录不按关键字顺序排列的文件。
  - 稠密索引：由于数据文件中记录不按关键字顺序排列，则必须对每个记录都建立一个索引项，如此建立的索引表称为稠密索引。
  - 非稠密索引：数据文件中的记录按关键字顺序有序，则可对一组记录建立一个索引项，这种索引表称为非稠密索引。

7.3 索引文件

- 索引文件的建立：
  - 索引表是由系统程序自动生成的
  - 索引文件的建立过程：
    - （1）按输入记录的先后次序建立数据区和索引表。其中索引表中关键字是无序的。
    - （2）待全部记录输入完毕后对索引表进行排序，排序后的索引表和主文件一起就形成了索引文件。

物理记录号	职工号	姓名	职务	其他
101	29	张三	程序员	.
103	05	李四	维护员	.
104	02	王五	程序员	.
105	38	刘六	测试员	.
108	31	.	.	.
109	43	.	.	.
110	17	.	.	.
112	48	.	.	

(a) 文件数据区

关键字	物理地址号
29	101
05	103
02	104
38	105
31	108
43	109
17	110
48	112

(b)输入过程中建立的索引表

	关键字	物理地址号
1	02	104
	05	103
	17	110
2	29	101
	31	108
	38	105
3	43	109
	48	112

(c) 索引表（排序后）

7.3 索引文件

索引文件的检索

- 检索方式：直接存取或按关键字（进行简单询问）存取
- 检索过程：
  - 首先，查找索引表
  - 若索引表上存在该记录，则根据索引项的指示读取外存上该记录；
  - 否则说明外存上不存在该记录，也就不需要访问外存。
- 由于索引项的长度比记录小得多，通常可将索引表一次读入内存，因此，在索引文件中进行检索只访问外存两次，即一次读索引，一次读记录。并且由于索引表是有序的，则查找索引表时可用折半查找法。

7.3 索引文件

索引文件的修改

- ①删除操作：删除一个记录时，仅需删去相应的索引项；
- ②插入操作：插入一个记录时，应将记录置于数据区的末尾，同时再索引表中插入索引项；
- ③更新操作：更新记录时，应将更新后的记录置于数据区末尾，同时修改索引表中相应的索引项。

多级索引

- 查找表：对索引表建立的索引。

最大关键字	物理块号
17	1
38	2
46	3

(d) 索引表的索引

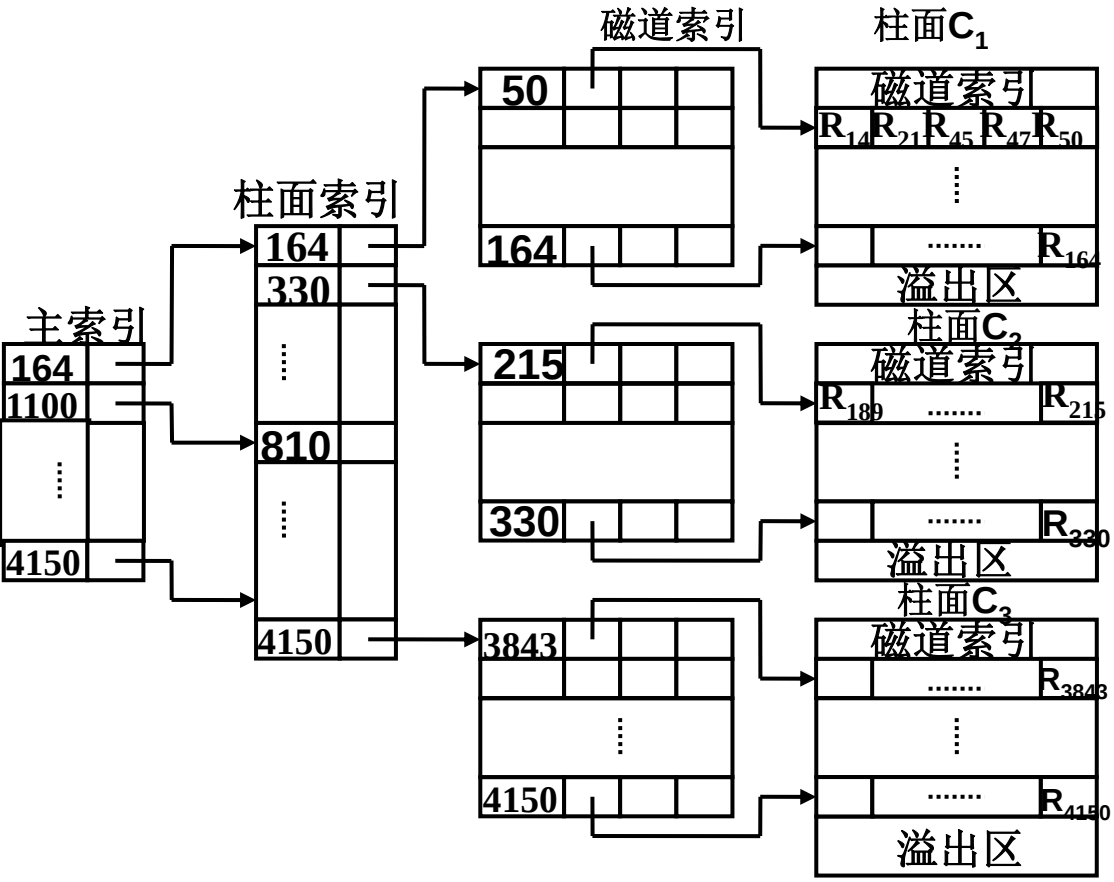
- 例如，假设图 (c) 的索引表需占3个物理块的外存，每一个物理块容纳3个索引，则建立的查找表如图 (d) 所示。
- 检索记录时，先查找查找表，再查索引表，然后读取记录
- 通常最高可有四级索引
  - 数据文件→索引表→查找表→第二查找表→第三查找表

7.3 索引文件

- 索引文件的组织---静态索引和动态索引
  - 索引文件的关键是如何组织索引。
  - 索引本身可以是顺序结构也可以是树型结构。由于大型文件的索引都相当大，则对顺序结构的索引需要建立多级索引。
    - 而树型结构本身就是一种“层次”结构，因此常用以作为索引文件的索引，如大型索引文件的索引--B树。
  - 多级索引是一种静态索引，为顺序表结构，虽然结构简单，但修改很不方便，每次修改都要重组索引。
    - 所以，当主文件在使用过程中记录变动比较多时，应采用树表结构的动态索引，如BST（AVL）、B-树、键树等，以便于插入、删除等。

7.4 ISAM文件

- 索引顺序存取方法ISAM(Indexed Sequential Access Method)
  - 定义：该方法是一种专为磁盘存取设计的文件组织方式。对磁盘上的数据文件建立盘组、柱面和磁道三级索引。



(1)磁道索引项：

①基本索引项：

关键字：表示该磁道中最末一个记录的关键字(在此为最大关键字)。

指针：指示该磁道中第一个记录的位置。

②溢出索引项：

关键字：表示该磁道溢出的记录的最大关键字。

指针：指示在溢出区中的第一个记录。

(2)柱面索引项：

关键字：表示该柱面中最末一个记录的关键字（最大关键字）。

指针：指示该柱面上的磁道索引位置

7.4 ISAM文件

- ISAM文件的检索：
  - 主索引→(柱面索引的索引)→，柱面索引→记录所在柱面的磁道索引→记录所在磁道的第一个记录的位置；
  - 由此出发在该磁道上进行顺序查找直至找到为止；或者找遍该磁道而不存在此记录，则表明该文件中无此记录.
- 溢出区和插入操作
  - 溢出区是为插入记录所设置的。每个柱面的基本区是顺序存储结构，而溢出区是链表结构。同一磁道溢出的记录由指针相链。
  - 溢出区的设置方法：
    - ①集中存放：整个文件设一个大的单一的溢出区。
    - ②分散存放：每个柱面设一个溢出区。上页例子即为此设置法。
    - ③集中与分散相结合：溢出时记录先移至每个柱面各自的溢出区，待满之后再使用公共溢出区。

7.4 ISAM文件

- 删除操作
  - ISAM文件中删除记录，只需找到待删除的记录，在其存储位置上做删除标记即可，而不需要移动记录或改变指针。
  - 但经过多次的增删后，文件的结构可能变得很不合理。此时，大量的记录进入溢出区，而基本区中又浪费很大空间.
  - 通常需要周期地整理ISAM文件。把记录读入内存，重新排列，复制成一个新的ISAM文件，填满基本区而空出溢出区
- 柱面索引的位置
  - 假设文件占有n个柱面，柱面索引在第x柱面上，则磁头移动距离的平均值为:

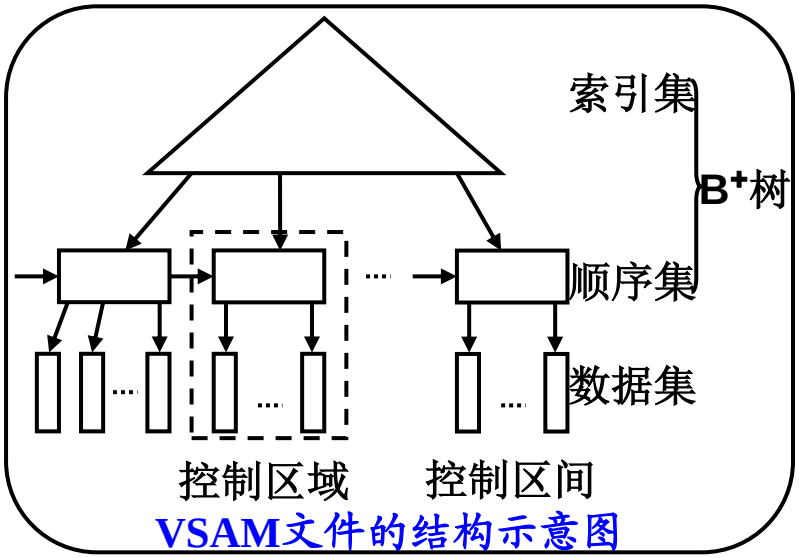
$$\bar{s} = \frac{1}{n}[\sum_{i=1}^{x}(x-i) + \sum_{i=x+1}^{n}(i-x)]$$

$$= \frac{1}{n}[x^2 - (n+1)x + \frac{n(n+1)}{2}]$$

  - 令 $\frac{d\bar{s}}{dx} = 0$，得到，$x = \frac{n+1}{2}$ 。这就是说，柱面索引应放在数据文件的中间位置的柱面上。

7.4 VSAM文件

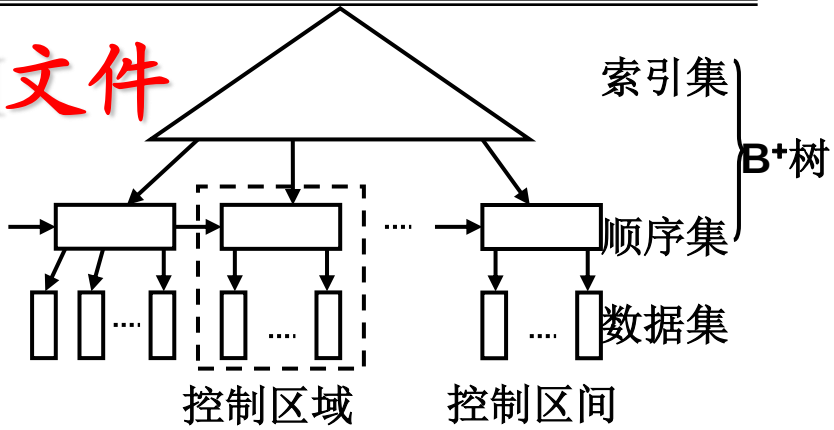
- 虚拟存储存取方法VSAM(Virtual Storage Access Methed)
  - 定义：该方法利用了操作系统的虚拟存储器的功能，给用户提供方便。对用户来说，文件只有控制区间和控制区域等逻辑存储单位，与外存储器中柱面、磁道等具体存储单位没有必然的联系。
  - 控制区间（Control Interval）：它是一个I/O操作的基本单位，由一组连续的存储单元组成。同一文件上控制区间的大小相同。
  - 控制区域（Control Range）：顺序集中一个结点连同对应所有控制区间形成的一个整体。



VSAM文件的结构示意图

  - 每个控制区间可视为一个逻辑磁道，而每个控制区域可视为一个逻辑柱面。

7.4 VSAM文件



VSAM文件的结构

- VSAM文件的结构由3部分组成：索引集、顺序集和数据集
- 文件的记录均存放在数据集中。顺序集和索引集一起构成一棵B^+树，为文件的索引部分。
- 顺序集中存放每个控制区间的索引项。每个控制区间的索引项由两部分信息组成，即该控制区间中最大关键字和指向控制区间的指针。若干相邻控制区间的索引项形成顺序集中的一个结点，结点之间用指针相链结，而每个结点又在其上一层的结点中建有索引，且逐层向上建立索引。
- 所有的索引项都由最大关键字和指针两部分信息组成，这些高层的索引项形成B^+树的非终端结点。因此，VSAM文件既可在顺序集中进行顺序存取，又可从最高层的索引（B^+树的根结点）出发进行按关键字存取。

7.4 VSAM文件

- 控制区间的结构:
  - 在VSAM文件中，记录可以是不定长的，则在控制区间中除了存放记录本身以外，还有每个记录的控制信息和整个区间的控制信息。
  - 在控制区间上存取一个记录时需从控制区间的两端出发同时向中间扫描。
  - 控制区间的结构示意图:

记录1	.....	记录n	未利用的空闲空间	记录n的控制信息	.....	记录1的控制信息	控制空间的控制信息

删除操作

- VASM文件删除记录时，需将同一控制区间中较删除记录关键字大的记录向前移动，把空间留给以后插入的新记录
- 若整个控制区间变空，则需修改顺序集中相应的索引项。

7.4 VSAM文件

插入操作与溢出处理

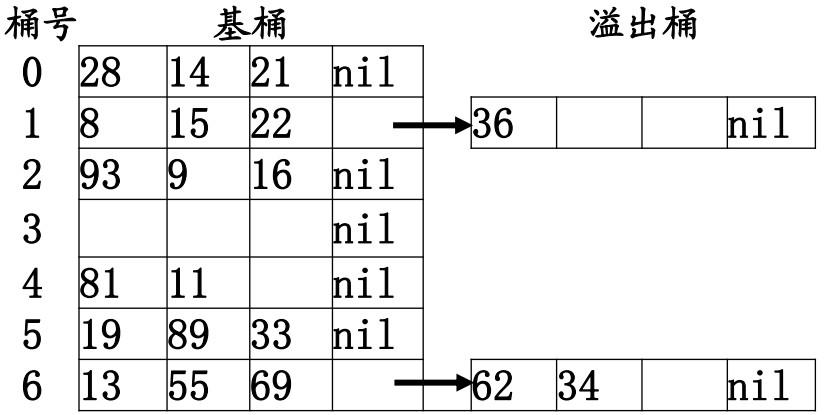
- VASM文件没有专门的溢出区，但可以利用控制区间中的空隙或控制区域中的空控制区间来插入记录（即在B+树上插入记录）。

VASM文件的特点

- 缺点：占有较多的存储空间，一般只能保持约75％的存储空间利用率。
- 优点：动态地分配和释放存储空间，无需像ISAM文件那样定期重排文件，并能较快地执行插入操作和保持较高的检索效率。
  - VSAM文件通常作为组织大型索引顺序文件的标准形式。

7.5 直接存取文件(散列文件)

- 定义
  - 直接存取文件是利用Hash法进行组织的文件。
  - 类似于Hash表，即根据文件关键字的特点设计Hash函数和处理冲突的方法将记录散列到外部存储设备上，故又称散列文件。



直接存取文件示例

- 溢出处理
  - “基桶”+“溢出桶”
  - 例如某一个文件有个记录，其关键字分别为28，19，13，93，89，14，55，69，8，9，15，16，22，33，81，62，l1，34，36，用除留余数法作哈希函数H（key）=key %7，桶的容量m=3，基本桶数=7，由此得到的散列文件如上图所示。

7.5 直接存取文件(散列文件)

- 散列文件的查找操作
  - （1）根据给定值求出散列地址（即基桶号）；
  - （2）将基桶的记录读入内存，进行顺序查找，若找到关键字等于给定值的记录，则检索成功；
  - （3）否则，若基桶内没有填满记录或其指针域为空，则文件内不含有待查记录；
  - （4）否则，根据指针域的值的指示将溢出桶的记录读入内存继续进行顺序查找，直至检索成功或不成功。
- 散列文件的删除操作
  - 删去一个记录，仅需对被删记录作删除标记即可。
- 散列文件的特点
  - 优点：文件随机存放，记录不需进行排序；插入、删除方便，存取速度快，不需要索引区，节省存储空间。
  - 缺点：不能进行顺序存取，只能按关键字随机存取；询问方式限于简单询问；在经过多次插入、删除后，可能造成文件结构不合理，需要重新组织文件。

7.6 多关键字文件

- 特点：在对文件进行检索操作时，不仅对主关键字进行简单询问，还经常需要对关键字进行其他类型的询问检索。

多重表文件(Multilist File)

- 特点：
  - ① 记录按主关键字的顺序构成一个串联文件，并建立主关键字的索引（称为主索引）。
  - ② 对每一个次关键字项建立次关键字索引（称为次索引），所有具有同一次关键字的记录构成一个链表。
  - ③ 主索引为非稠密索引，次索引为稠密索引。每个索引项包括次关键字、头指针和链表长度。
- 示例：

7.6 多关键字文件(cont.)

- (a)所示为一个多重表文件。其中，学号为主关键字，记录按学号顺序链接，为了查找方便，分成3个子链表，其索引如(b) 所示，索引项中的主关键字为各子表中的最大值。
- 专业、已修学分和选修课程为3个次关键字项，它们的索引(c)-(e)所示，具有相同次关键字的记录链接在同一链表中。

记录号	姓名	学号		专业		已修学分		选修课程					
01	王雯	1350	02	软件	02	412	03	丙	02	丁	03		
02	马雁	1351	03	软件	07	398	07	甲	04	丙	03		
03	阮森	1352	04	计算机	05	436	nil	乙	05	丙	04	丁	05
04	苏明	1353	nil	应用	06	402	08	甲	06	丙	08		
05	田勇	1354	06	计算机	nil	384	02	乙	07	丁	09		
06	杨青	1355	07	应用	09	356	10	甲	07				
07	薛平	1356	08	软件	08	398	nil	甲	08	乙	nil		
08	崔键	1357	nil	软件	nil	408	01	甲	09	丙	nil		
09	王洪	1358	10	应用	10	370	05	甲	10	丁	nil		
10	刘倩	1359	nil	应用	nil	364	09	甲	nil				

(a) 数据文件

主关键字	头指针
1353	01
1357	05
1359	09

(b) 主关键字索引

次关键字	头指针	长度
软件	01	4
计算机	03	2
应用	04	4

(c) “专业” 索引

次关键字	头指针	长度
350-399	01	4
400-499	03	2

(d) “已修学分” 索引

次关键字	头指针	长度
甲	02	7
乙	03	3
丙	01	5
丁	01	4

(e) “选修课程” 索引

7.6 多关键字文件(cont.)

倒排文件(Inverted index)

- 倒排文件和多重表文件的次关键字索引的结构不同。
- 在倒排表中，具有相同次关键字的记录不设指针相链，而是在次关键字的一项中存放这些记录的物理记录号。
- 通常称倒排文件中的次关键字索引为倒排表。

基于属性的倒排表

记录号	姓名	学号		专业		已修学分		选修课程					
01	王雯	1350	02	软件	02	412	03	丙	02	丁	03		
02	马雁	1351	03	软件	07	398	07	甲	04	丙	03		
03	阮森	1352	04	计算机	05	436	nil	乙	05	丙	04	丁	05
04	苏明	1353	nil	应用	06	402	08	甲	06	丙	08		
05	田勇	1354	06	计算机	nil	384	02	乙	07	丁	09		
06	杨青	1355	07	应用	09	356	10	甲	07				
07	薛平	1356	08	软件	08	398	nil	甲	08	乙	nil		
08	崔键	1357	nil	软件	nil	408	01	甲	09	丙	nil		
09	王洪	1358	10	应用	10	370	05	甲	10	丁	nil		
10	刘倩	1359	nil	应用	nil	364	09	甲	nil				

(a) 数据文件

软件	01, 02, 07, 08
计算机	03, 05
应用	04, 06, 09, 10

(f) “专业”倒排表

350-399	02, 05, 06, 07, 09, 10
400-499	01, 03, 04, 08

(g) “已修学分”倒排表

甲	02, 04, 06, 07, 08, 09, 10
乙	03, 05, 07
丙	01, 02, 03, 04, 08
丁	01, 03, 05, 09

(h) “选修课程”倒排表

7.6 多关键字文件(cont.)

- 倒排索引的优点
  - 检索记录较快。特别是对某些询问，不用读取记录，就可得到回答，如询问“软件”专业的学生中有否选课程“乙”的，则只要将“软件”索引中的记录号和“乙”索引中的记录号作求“交”即可。
- 插入和删除
  - 在插入和删除记录时，倒排表也要作相应的修改。因为倒排表中具有同一次关键字的记录号是有序排列的，所以修改时要作相应移动。
- 倒排文件的缺点
  - 维护困难。在同一索引表中，不同的关键字其记录数不同，各倒排表的长度不等，同一倒排表中各项长度也不等。
- 若数据文件非串链文件，而是索引顺序文件(如ISAM文件)，则倒排表中应存放记录的主关键字而不是物理记录号。

7.6 多关键字文件(cont.)

基于文本的倒排索引

- 也可以应用于非结构化的信息检索，如大量正文的文本索引。尤其当今搜索引擎需要对海量的正文文本信息进行检索的情况下，倒排文件的使用尤其重要。
- 对多个正文文本建立索引的基本思想：把正文看成一个一个的关键词的集合，然后用这些词组成一些适合快速检索的数据结构。一个倒排文件就是一个已经排好序的关键词的列表，其中每个关键词指向一个倒排表，该表中记录了该关键词出现的文档集合以及在该文档中的出现位置。如某图书馆论文集的部分倒排表：

关键词	倒排表(所在文档编号，出现次数，出现位置)
KMP	（#3307, 2, 5, 43）（#4615, 5, 0, 19, 34, 70, 143）……
MST	（#2519, 1, 267）（#6742, 3, 19, 322, 526）……
B-Tree	（#2948, 3, 45, 267, 587）（#3693, 5, 39, 423, 765,809,1024）……
……	……

文本倒排表

7.7 磁盘文件的归并排序

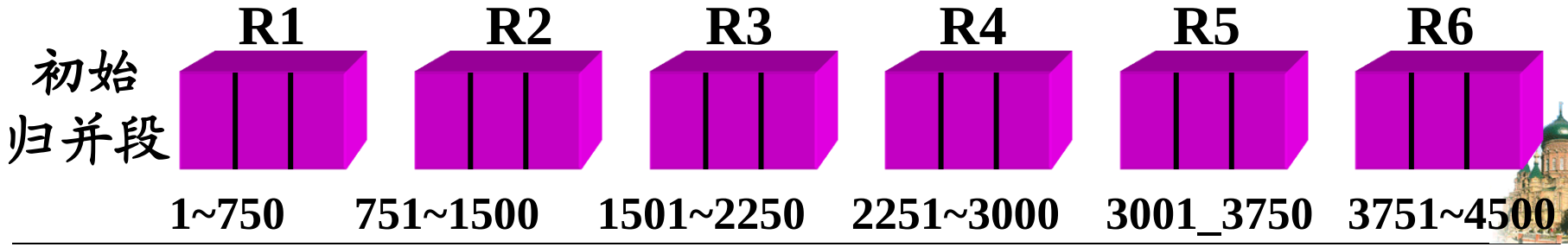
- 外部排序的概念
  - 是指在排序的过程中，数据的主要部分存放在外存储器上，借助内存储器(作为工作单元)，来调整外存储器上数据的位置。
- 外部归并排序重点研究的问题
  - 如何进行多路归并以减少文件的归并遍数;
  - 如何巧妙地运用内存的缓冲区使I/O和CPU尽可能并行工作
  - 根据外存的特点选择较好的产生初始归并段的方法。
- 分类:
  - 磁盘和磁带归并排序
  - 磁盘是随机存储设备
  - 磁带是顺序存储设备

7.7 磁盘文件的归并排序

- 外部排序归并方法的一般过程：分两个阶段
  - 第一阶段：初始归并段形成
    - 首先，将文件中的数据分段输入到内存，在内存中采用内部排序方法对其进行排序（排序完的文件段，称为归并段run），
    - 然后将有序段写回外存。
    - 整个文件经过在内存逐段排序又逐段写回外存，这样在外存中形成多个初始的归并段。
  - 第二阶段：多路归并
    - 对这些初始归并段采用某种归并排序方法，进行多遍归并，最后形成整个文件的单一归并段(整个文件有序)。

7.7 磁盘文件的归并排序

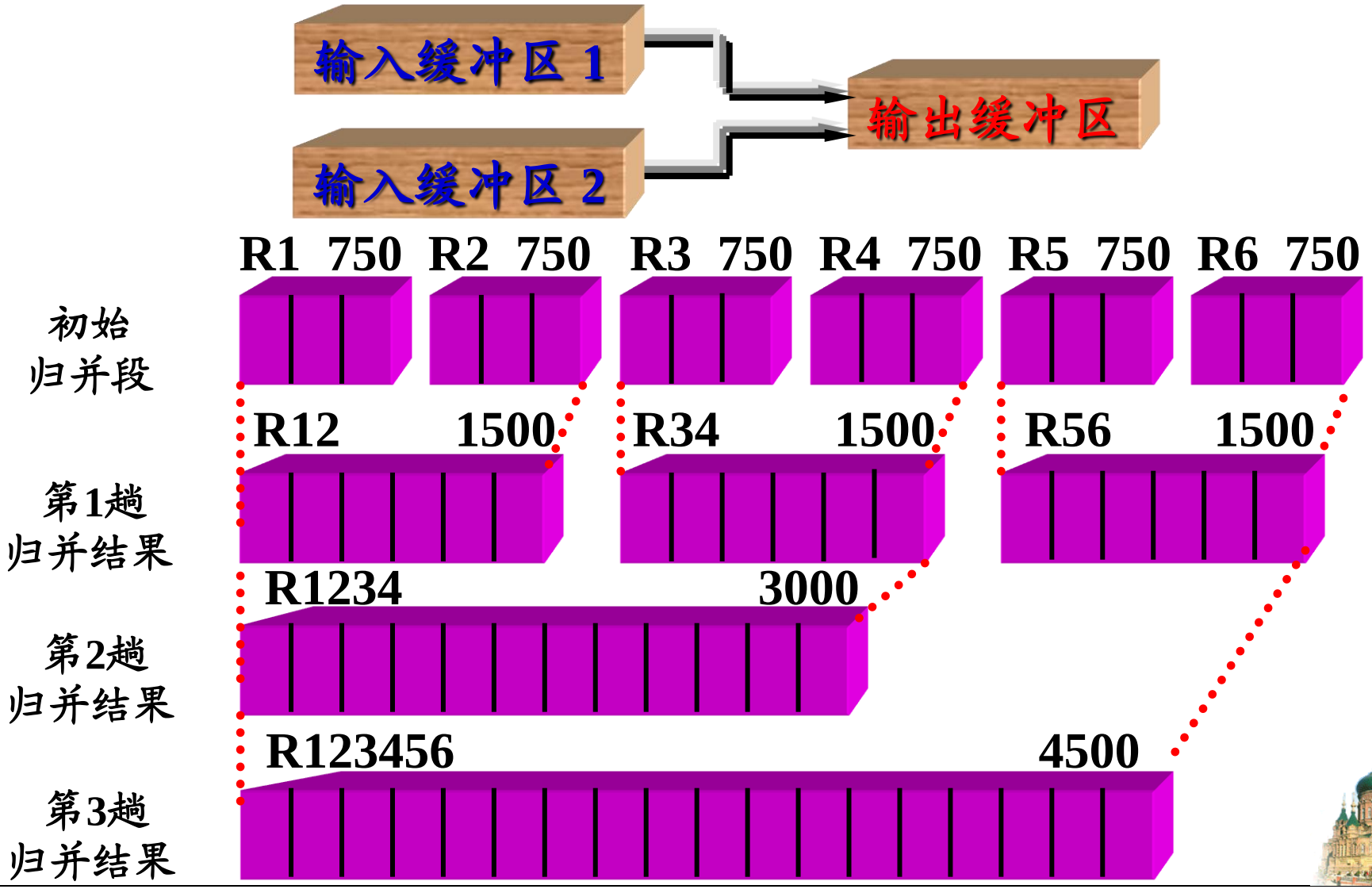
- 示例：设有一个包含4500个记录的输入文件。现用一台其内存至多可容纳750个记录的计算机对该文件进行排序。输入文件放在磁盘上，磁盘每个页块可容纳250个记录，这样全部记录可存储在 4500 / 250 = 18 个页块中。输出文件也放在磁盘上，用以存放归并结果。
- 由于内存中可用于排序的存储区域能容纳750 个记录，因此内存中恰好能存3个页块的记录。
- 在外排序一开始，把18块记录，每3块一组，读入内存。利用某种内排序方法进行内排序，形成初始归并段，再写回外存。总共可得到6个初始归并段。然后一趟一趟进行归并排序。



7.7 磁盘文件的归并排序

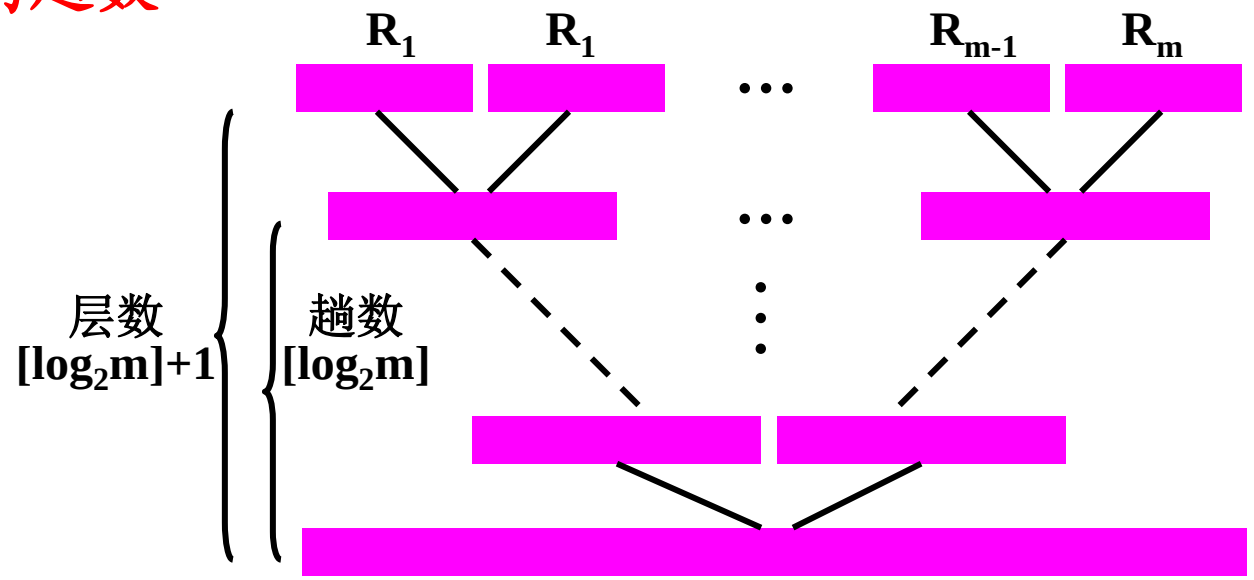
- 若把内存区域等份地分为 3 个缓冲区。其中的两个为输入缓冲区, 一个为输出缓冲区, 可以在内存中利用简单 2 路归并函数 MergeSort( ) 实现 2 路归并。
- 首先, 从参加归并排序的两个输入归并段 *R*1 和 *R*2 中分别读入一块, 放在输入缓冲区1 和输入缓冲区2 中。然后在内存中进行 2 路归并，归并结果顺序存放到输出缓冲区中。
- 当输出缓冲区装满250个记录时，就输出到磁盘。
- 如果归并期间某个输入缓冲区空了，就立即向该缓冲区继续装入所对应归并段的一块记录信息，使之与另一个输入缓冲区的剩余记录归并，直到R1和R2归并为R12、 R3和R4归并为R34、R5和R6归并为R56为止。
- 再把R12和R34归并为R1234，最后把R1234和R56归并为R123456（如下页图示）

7.7 磁盘文件的归并排序



7.7 磁盘文件的归并排序

归并的趟数



m 个归并段的归并过程

- m个初始段，进行2路归并，需要⌈$\log_2 m$⌉趟归并；
- m个初始段，采用K路归并，需要⌈$\log_K m$⌉趟归并。
- 显然，K越大，归并趟(遍)数越少，可提高归并的效率。

7.7 磁盘文件的归并排序

- (1)多路归并—可以减少归并趟数，但可能增加比较次数
  - 在 K路归并时，从 K 个关键字中选择最小记录时，要比较K-1 次。若记录总数为 n ，每趟要比较 n*(K-1)次，$\lceil \log_K m \rceil$趟要比较的次数为:

    $$n*(K-1)\lceil \log_K m \rceil = n*(K-1)\lceil \log_2 m/\log_2 K \rceil$$

  - 可以看出，随着K增大，$(K-1)/\log_2 K$ 也增大，当归并路数多时， CPU 处理的时间(比较次数)也随之增多。当K值增大到一定程度时，可能使CPU处理时间大于因K值增大而减少归并趟数所节省的时间！
  - 为此可以
    - ①选择好的(归并)排序方法，以减少排序中比较次数
    - ②选择好的初始归并段形成方法，增大归并段长度 (亦即减少初始归并段数量)，从而提高排序的效率。

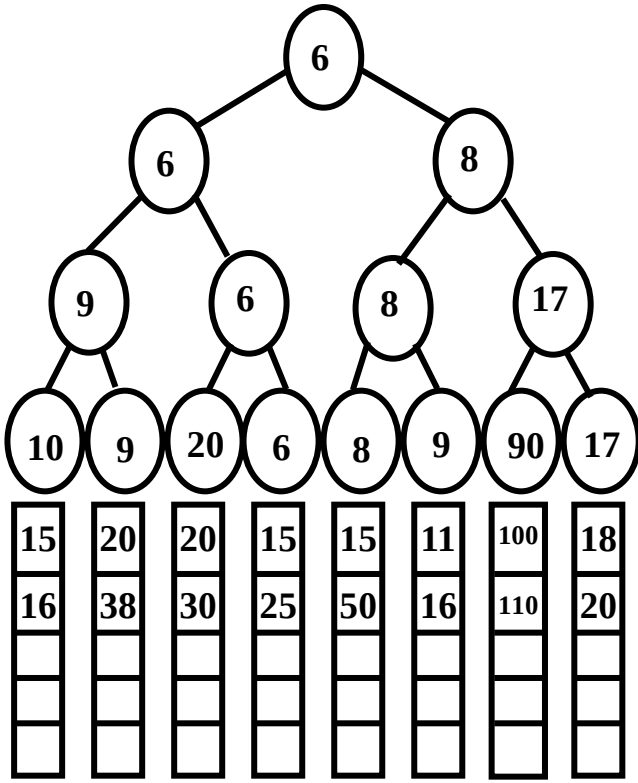
7.7 磁盘文件的归并排序

- (2)K 路平衡归并与败者树—置换-选择
  - 第一次建立选择树的比较所花时间为:

    $O(K-1) = O(K)$
  - 而后，每次重建选择树所需时间为:
    $O(\log_2 K)$
  - n 个记录处理时间为初始建立选择树的时间加上 n-1 次重建选择树的时间

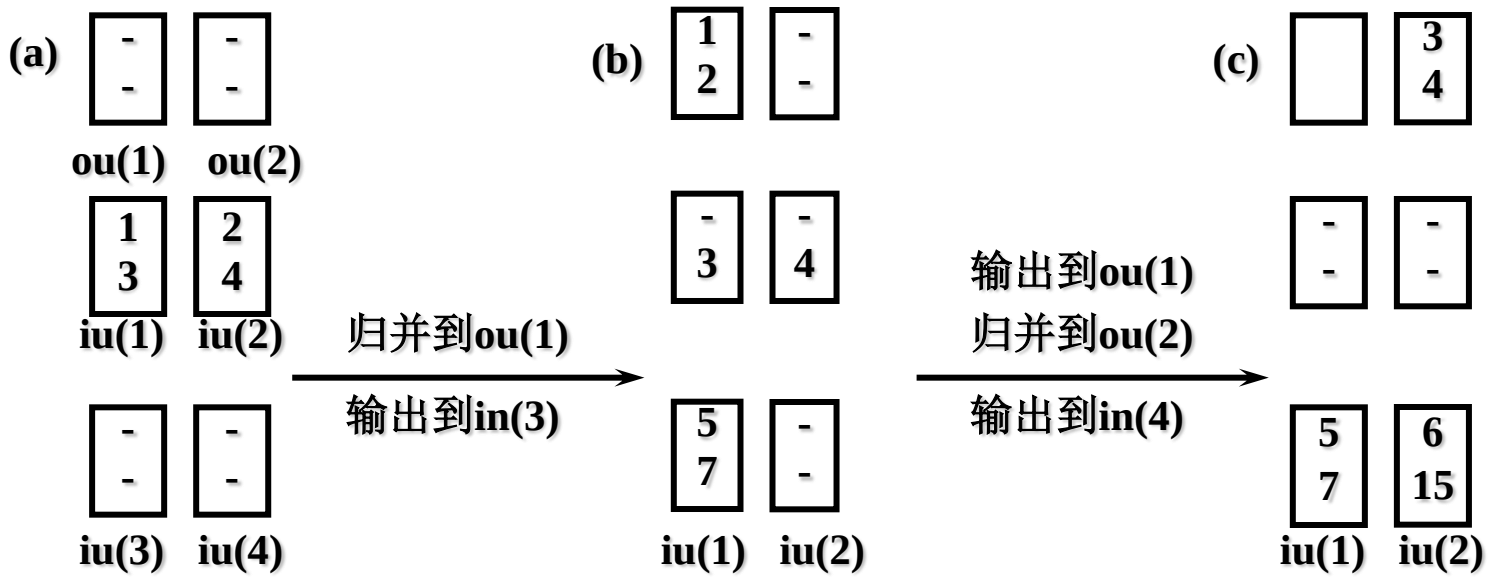
    $O((n-1)\cdot\log_2 K)+O(K) = O(n\cdot\log_2 K)$
  - 这就是K路归并一趟所需的CPU处理时间。归并趟数为 $\log_K m$，总时间为

    $O(n \log_2 K \log_K m) = O(n \log_2 m)$
  - K 路归并 的CPU 时间与 K 无关---选择树太好了！



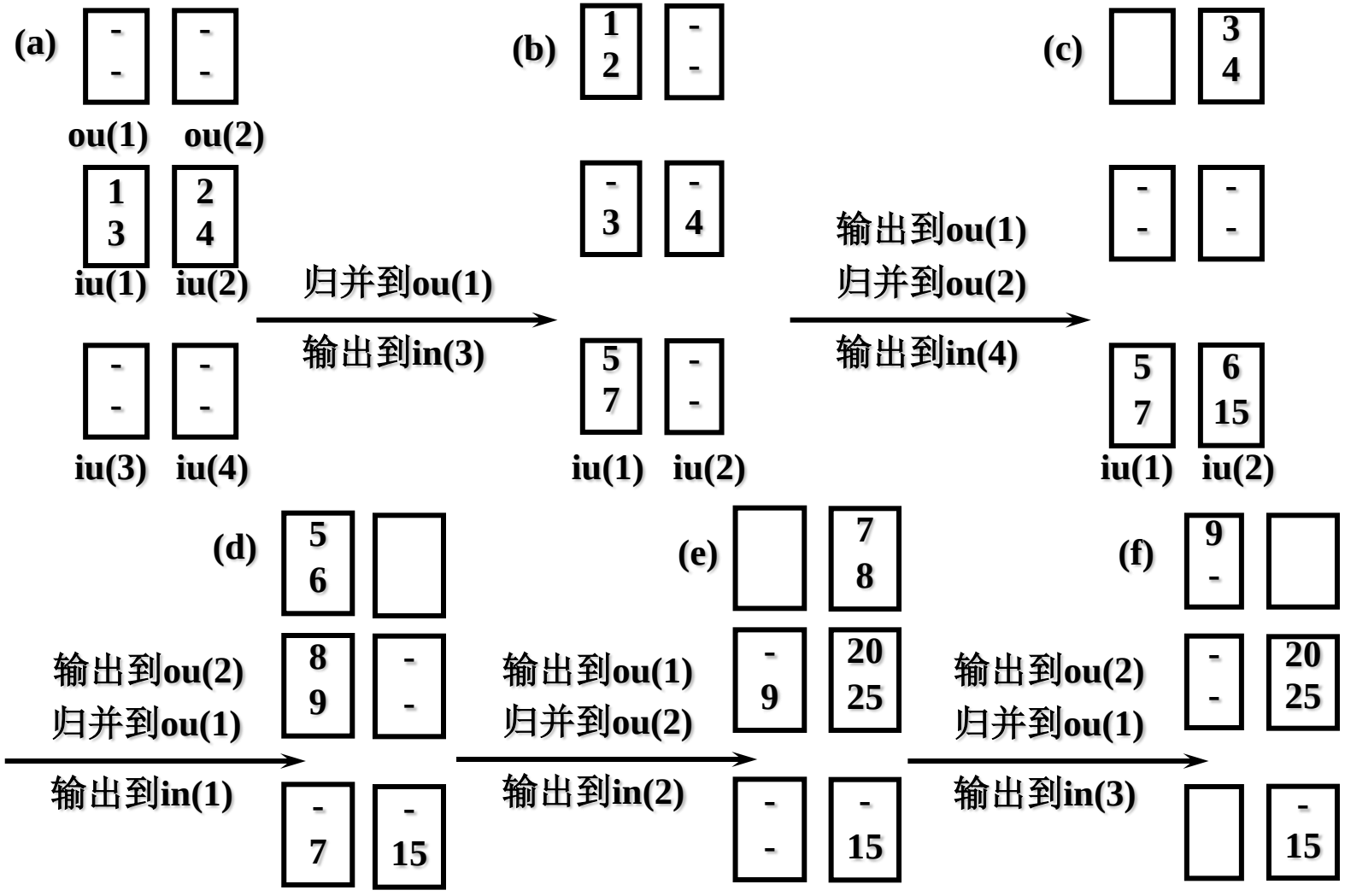
7.7 磁盘文件的归并排序

- (3) 并行操作的缓冲区处理——使I/O和 CPU 处理尽可能重叠
  - 对K个归并段进行 K 路归并至少需要K个输入和1个输出缓冲区。要使输入、输出和归并同时进行，K+1个缓冲区是不够的，需要2(K+1)个缓冲区实现并行操作。



7.7 磁盘文件的归并排序

- 并行操作的缓冲区处理----使I/O和 CPU 处理尽可能重叠



7.7 磁盘文件的归并排序

- (4)初始归并段的生成--尽量增加初始归并段的长度，从而减少个数
  - 任何内部排序算法都可作为生成初始归并段的算法
  - 初始归并段的长度 ≥ 缓冲区的长度？！
- 选择树法—置换选择排序
- 假设初始待排序文件为输入文件FI，初始归并段文件为输出文件FO，内存缓冲区为W，可容纳P个记录。FO，W初始为空，则置换-选择如下：
  - (1) 从FI输入P个记录到缓冲区W;
  - (2) 从W中选择出关键字最小的记录MIN;
  - (3) 将MIN记录输出到FO中去;
  - (4) 若FI不空，则从FI输入下一个记录到W;
  - (5)从W中所有比MIN关键字大的记录中选出最小关键字记录，作为新的MIN;(W中比MIN关键字小的暂时不能选，即不能作为当前归并段的记录)
  - (6)重复(3)~(5)，直到在W中选不出新的MIN为止。得到一个初始归并段，输出归并段结束标志到FO中
  - (7)重复(2)~(6)，直到W为空，由此得到全部初始归并段。

7.7 磁盘文件的归并排序

- 示例：缓冲区W的长度P=4，输入序列为：

15 19 04 83 12 27 11 25 16 34 26 07 10 90 06 …

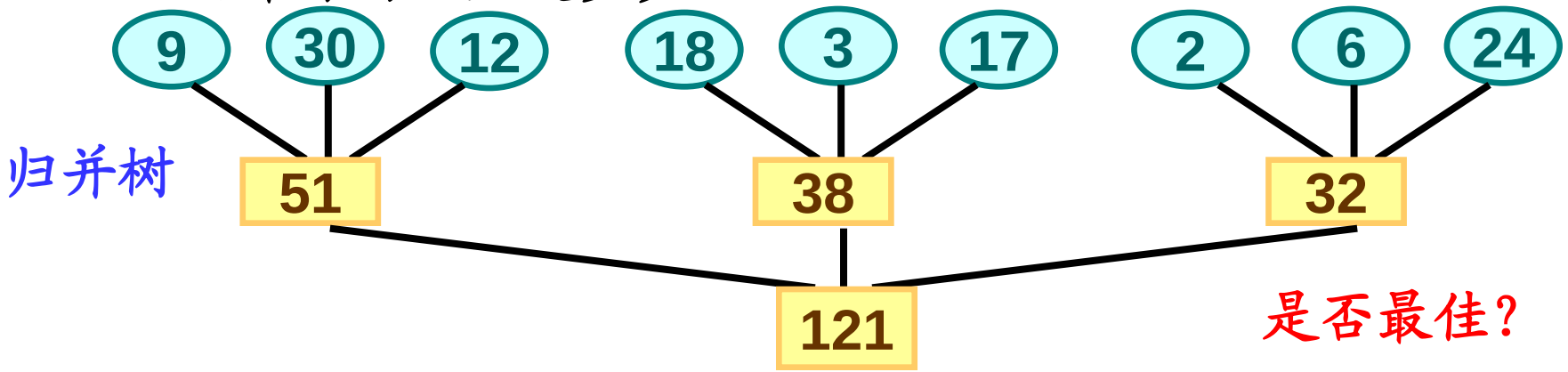
注意:如果新输入记录的关键字小于最后输出记录的关键字，则新输入记录不能成为当前归并段的一部分；它要等待生成下一个归并段时供选择。

步	1	2	3	4	5	6	7	8	9	10	11	12	13	…
缓冲区内容	15	15	15	(11)	(11)	(11)	(11)	(11)	(11)	11	11	(06)	…	…
	19	19	19	19	25	(16)	(16)	(16)	(16)	16	16	16	…	…
	04	12	27	27	27	27	34	(26)	(26)	26	26	26	…	…
	83	83	83	83	83	83	83	83	(07)	10	90	90	…	…
输出结果	04	12	15	19	25	27	34	83	07	10	11	16	…	
	R_1								R_2					

- 采用选择树法生成初始归并段的平均长度是缓冲区长度的两倍。

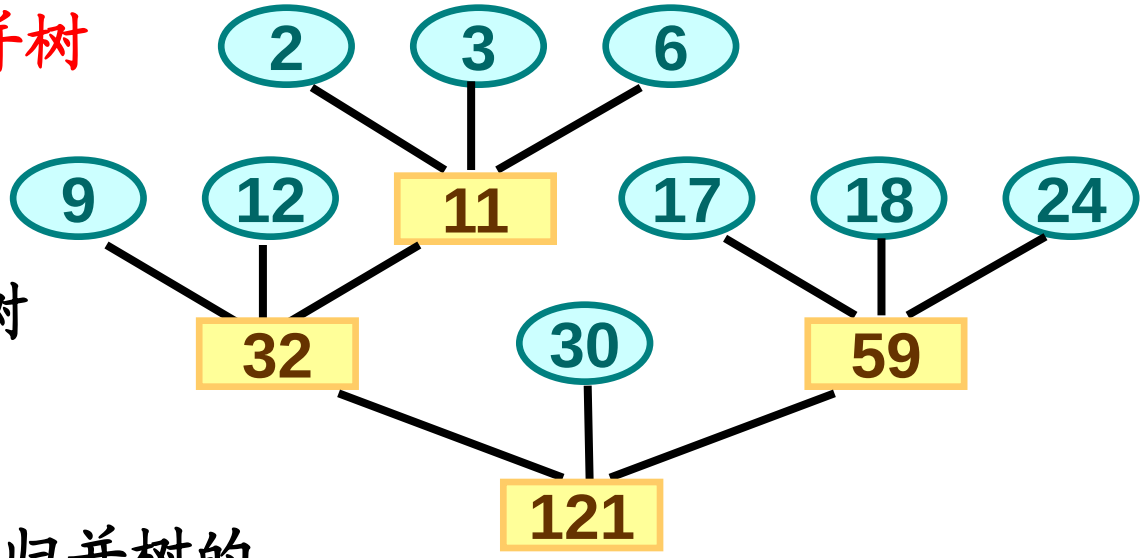
7.7 磁盘文件的归并排序

- (5)最佳归并树——使外存读写次数最少
  - 由置换-选择排序所得初始归并段的长度可能不等，这对于多路平衡归并将产生什么影响？
  - 例：假设经置换-选择排序先后得到的归并段长度分别为：
    9，30，12，18，14，25，31，7，27
    且每个记录占一个物理块，则进行3-路平衡归并排序时访问外存的次数是多少？



  - 三叉树的带权路径长度：
    WPL=（9+30+12+18+3+17+2+6+24）× 2 = 242
  - 访问外存次数：242 × 2 = 484 (读/写各1次)

最佳归并树



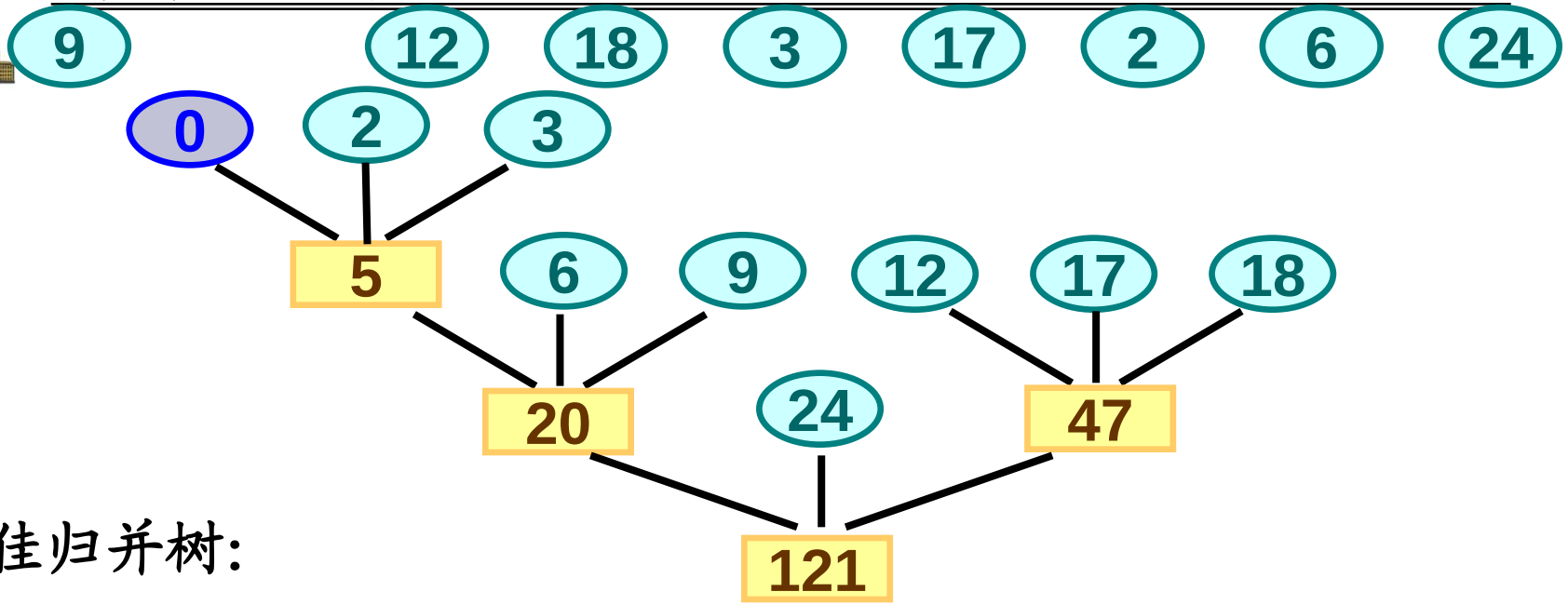
最佳归并树

- 最佳归并树的
  - 带权路径长度:

$$WPL=(2+3+6)\times3+(9+12+17+18+24)\times2+30\times1=223$$

  - 访问外存次数: $223\times2=446$ (读/写各1次)

- 如果去掉一个长度为30的初始归并段，求最佳归并树?
  - $WPL=(2+3+5)\times3+(21+59)\times2=193$
  - 访问外存次数: $193\times2=386$ 是否有更少的?



- 最佳归并树:
  - WPL = (2+3)×3+(15+47)×2+24 ×1 = 163
  - 访问外存次数: 163 × 2 = 326
- 问题: 当初始归并段的数目不足时，怎样求最佳归并树?
  - 最佳归并树应该是一棵“正则树”
  - 对 K 路归并而言，设初始归并段为 m，若:

$$(m-1)\%(K-1)=0$$

则不需要加虚段，否则需要加虚段的个数为:

$$K-(m-1)\%(K-1)-1$$

7.8 磁带文件的归并排序

- 与磁盘不同，磁带是顺序存储设备，读取信息块的时间与信息块的位置有关。研究磁带排序，需要了解信息块的分布。
- K路平衡归并排序
  - 磁带机数量：2K

输入：T_1, T_2, ……, T_k　　输出

输出：T_{k+1}, T_{k+2}, ……, T_{2k}　　输入

磁带机	T_1	T_2	…	T_k
归并段	R_1	R_2	…	R_k
	R_{k+1}	R_{k+2}	…	R_{2k}
	…	…	…	…
	R_{mk+1}	…	…	…

T_1:R_1(1000),R_3(1000),R_5(1000)
T_2:R_2(1000),R_4(1000),R_6(1000)
T_3:Ø
T_4:Ø

T_1: Ø
T_2: Ø
T_3: R_1(2000),R_3(2000)
T_4: R_2(2000)

T_1: R_1(4000)
T_2: R_2(2000)
T_3: Ø
T_4: Ø

→

T_1: Ø
T_2: Ø
T_3: R_1(6000)
T_4: Ø

7.8 磁带文件的归并排序

多阶段归并排序

K+1台磁带机，实现 k 路归并

$$F_n^{(k)} = 0$$

$$F_n^{(k)} = 1$$

$$F_n^{(k)} = F_{n-1}^{(k)} + F_{n-2}^{(k)} + \cdots + F_{n-k}^{(k)}$$

$$t_1^j = F_{j+k-2}^{(k)}$$

$$t_2^j = F_{j+k-3}^{(k)} + F_{j+k-2}^{(k)}$$

$$\cdots$$

$$t_{k-1}^j = F_j^{(k)} + F_{j+1}^{(k)} + \cdots + F_{j+k-2}^{(k)}$$

$$t_k^j = F_{j-1}^{(k)} + F_j^{(k)} + \cdots + F_{j+k-2}^{(k)}$$

$$F_{G(j+k-2)}^{(k)} = t_1^j + t_2^j + \cdots + t_{j+k-2}^{(k)}$$

i 遍后	t_1	t_2	t_3
开始	13(1L)	21(L)	空
1	空	8(1L)	13(2L)
2	8(3L)	空	5(2L)
3	3(3L)	5(5L)	空
4	空	2(5L)	3(8L)
5	2(13L)	空	1(8L)
6	1(13L)	1(21L)	空
7	空	空	1(34L)

步	t_1	T_2	t_3	总段数
n	0	0	1	1
n-1	1	1	0	2
n-2	2	0	1	3
n-3	0	2	3	5
n-4	3	5	0	8
n-5	8	0	5	13
n-6	0	8	13	21
n-7	13	21	0	34

本章小结

- 基本概念
  - 文件、属性和域、关键字、主关键字、次关键字
- 文件的操作
  - 检索方式和更新方式
- 文件的组织方式包括那些？各有什么特点？适合什么场合？

本章小结

- 内部排序过程中不涉及数据的内、外存交换，待排序的记录全部存放在内存中；
- 若待排序的文件很大，就无法将整个文件的所有记录同时调入内存进行排序；
- 外部排序的实现，主要是依靠数据的内、外存交换和“内部归并”。
- 外部排序基本上包括相对独立的两个阶段：初始归并段的形成和多路归并。
- 外部排序主要研究的技术问题是：
  - 如何进行多路归并以减少文件的归并遍数；
  - 如何运用内存的缓冲区使I/O和CPU尽可能并行工作；
  - 根据外存的特点选择较好的产生初始归并段的方法。

本章小结

- 初始归并段的形成:
  - 目标：尽量增加初始归并段的长度，从而减少个数
  - 任何内部排序算法都可作为生成初始归并段的算法
  - 初始归并段的长度 ≥ 缓冲区的长度？！
  - 置换-选择排序，生成初始归并段
  - 采用选择树法(置换-选择排序)生成初始归并段的平均长度是缓冲区长度的两倍
- I/O和CPU尽可能并行工作
  - 进行 K 路归并至少需要K个输入和1个输出缓冲区。
  - 要使输入、输出和归并同时进行，K+1个缓冲区是不够的
  - 缓冲区备份法：需要2(K+1)个缓冲区实现并行操作

本章小结

- 多路归并: (假设有n个记录，m个初始段)
  - 采用2路归并，需要$\lceil \log_2 m \rceil$趟归并;
  - 采用K路归并，需要$\lceil \log_K m \rceil$趟归并。
    - 显然，K越大，归并趟数越少，可提高归并的效率
    - 但可能增加整个归并过程的比较次数:
    - $n*(K-1)\lceil \log_K m \rceil = n*(K-1)\lceil \log_2 m/\log_2 K \rceil$
- 多路平衡归并---选择树法
  - 建立选择树的时间：$O(K-1) = O(K)$
  - 重构选择树的时间：$O(\log_2 K)$
  - 一趟归并的时间：$O((n-1)\cdot\log_2 K)+O(K) = O(n\cdot\log_2 K)$
  - 总的归并时间：$O(n \log_2 K \log_K m) = O(n \log_2 m)$

本章小结

- 最佳归并树:
  - 由置换-选择排序所得初始归并段的长度可能不等
  - 目标：最小化多路平衡归并的外存读写次数
  - 当初始归并段的数目不足时，怎样进行最佳归并?
    - 最佳归并树应该是一棵“正则树”
    - 对 K 路归并而言，设初始归并段为 m，
      - 若(m－1) % (K－1) = 0，则不需要加虚段;
      - 否则，需要加虚段的个数为:
        - K－(m－1) % (K－1)－1